ICS第十二章

1. volatile保证定义的变量存放在内存中，而不总是在寄存器里。右侧为两个进程的地址空间。请在合适的位置标出变量gCount、vCount与lCount的位置。如果一个量出现多次，那么就标多次。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| long **gCount** = 0;  void \*thread(void \*vargp) {  volatile long **vCount** = \*(long \*)vargp;  static long **lCount** = 0;  gCount++; vCount++; lCount++;  printf("%ld\n", gCount+vCount+lCount);  return NULL;  }  int main() {  long var; pthread\_t tid1, tid2;  scanf("%ld", &var);  fork();  pthread\_create(&tid1,NULL,thread,&var);  pthread\_create(&tid2,NULL,thread,&var);  pthread\_join(tid1, NULL);  pthread\_join(tid2, NULL);  return 0;  } |  | vCount  vCount | 高地址 | vCount  vCount |
| ... |  | ... |
| 共享库 | 共享库 | 共享库 |
| ... |  | ... |
|  | 堆 |  |
| gCount  lCount | 数据区 | gCount  lCount |
|  | 代码区  低地址 |  |
| 父进程 |  | 子进程 |

2. 下面的程序会引发竞争。一个可能的输出结果为2 1 2 2。解释输出这一结果的原因。

|  |
| --- |
| long foo = 0, bar = 0;  void \*thread(void \*vargp) {  foo++; bar++;  printf("%ld %ld ", foo, bar); fflush(stdout);  return NULL;  }  int main() {  pthread\_t tid1, tid2;  pthread\_create(&tid1, NULL, thread, NULL);  pthread\_create(&tid2, NULL, thread, NULL);  pthread\_join(tid1, NULL);  pthread\_join(tid2, NULL);  return 0;  } |

【答】线程1将foo、bar改为1以后被线程2打断，线程2将foo改为2以后被线程1打断，线程1输出了2 1，线程2将bar改为2，并输出了2 2。

3. 信号量w,x,y,z均被初始化为1。下面的两个线程运行时可能会发生死锁。给出发生死锁的执行顺序。

|  |  |
| --- | --- |
| 线程1 | ①P(w) ②P(x) ③P(y) ④P(z) ⑤V(w) ⑥V(x) ⑦V(y) ⑧V(z) |
| 线程2 | ⅠP(x) ⅡP(z) ⅢP(y) ⅣP(w) ⅤV(x) ⅥV(y) ⅦV(w) ⅧV(z) |

【答】①→Ⅰ→Ⅱ→Ⅲ，此时线程1占用了w而在等待x，线程2占用了x而在等待w。

4. 某次考试有30名学生与1名监考老师，该教室的门很狭窄，每次只能通过一人。考试开始前，老师和学生进入考场（有的学生来得比老师早），当人来齐以后，老师开始发放试卷。拿到试卷后，学生就可以开始答卷。学生可以随时交卷，交卷后就可以离开考场。当所有的学生都上交试卷以后，老师才能离开考场。

请用信号量与PV操作，解决这个过程中的同步问题。所有空缺语句均为PV操作。

|  |
| --- |
| 全局变量：  stu\_count: int类型，表示考场中的学生数量，初值为0  信号量：  mutex\_stu\_count: 保护全局变量，初值为1  mutex\_door: 保证门每次通过一人，初值为**1**  mutex\_all\_present: 保证学生都到了，初值为**0**  mutex\_all\_handin: 保证学生都交了，初值为**0**  mutex\_test[30]: 表示学生拿到了试卷，初值均为**0** |

|  |  |
| --- | --- |
| Teacher: // 老师  **P(mutex\_door)**  从门进入考场  **V(mutex\_door)**  **P(mutex\_all\_present)** // 等待同学来齐  for (i = 1; i <= 30; i++)  **V(mutex\_test[i])** // 给i号学生发放试卷    **P(mutex\_all\_handin)** // 等待同学将试卷交齐  **P(mutex\_door)**  从门离开考场  **V(mutex\_door)** | Student(x): // x号学生  **P(mutex\_door)**  从门进入考场  **V(mutex\_door)**  P(mutex\_stu\_count);  stu\_count++;  if (stu\_count == 30)  **V(mutex\_all\_present)**  V(mutex\_stu\_count);  **P(mutex\_test[i])** // 等待拿自己的卷子  学生答卷  P(mutex\_stu\_count);  stu\_count--;  if (stu\_count == 0)  **V(mutex\_all\_handin)**  V(mutex\_stu\_count);  **P(mutex\_door)**  从门离开考场  **V(mutex\_door)** |